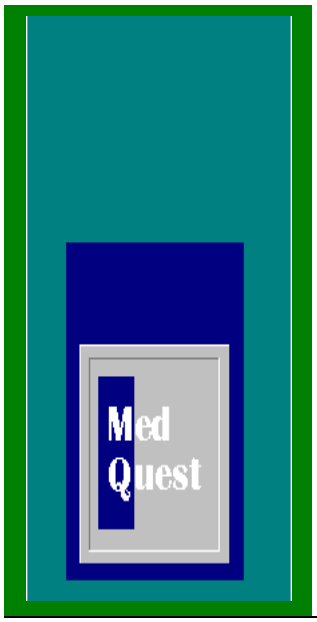


MEDQUEST SOFTWARE

RULE EDITORS: TECHNICAL PAPER # 4

September 30, 1998

TABLE OF CONTENTS

OVERVIEW

MedQuest Technical Paper 4: Rule Editor has been prepared to provide you with information about the functions in MedQuest. In the MedQuest design option, you have the ability to add different types of rules depending on the sequence of events occurring in data entry (e.g., during entry, after entry, on exiting the case, etc.). You can use the Rule Editor to program these rules. Types of rules you might want to build include warning the user if a value is outside of a given range or adding a rule to "skip" entry of a variable. This capability enables you to build an "expert" data entry system.

RELATIONSHIP TO SQL

The data entry rules created by the MedQuest user are not written using the structured query language (SQL), but are built using a special MedQuest rule syntax to optimize the performance speed during rule execution. However, the MedQuest rule syntax can be parsed to translate the logic into an equivalent SQL statement.

RELATIONSHIP TO DICTIONARY

The data entry rules created by the MedQuest Rule Editor are saved in the Equation table (EQUATIONS) of the dictionary database. This table is related to other tables by the **Fieldname**, **Actionfieldname**, and **Screenname** fields.

RULE EDITOR TERMINOLOGY

The **Rule Editor** is the MedQuest function that is used for creating and editing rules that execute during data entry. All of the logic in your data entry rules is built in the **Rule Editor**. There are a few terms that you should understand before you begin creating rules.

Rule

A set of logic that evaluates to true or false and is associated with a given event (e.g., on entry of a variable). Its value, if true, results in a given action (for example, disabling a variable). MedQuest can have many rules for many different events and for many different actions. A rule can be described by the following example:

On event X execute rule Y. If rule Y is true, then do action Z.

Logic

Each row of logic must contain a left side(cell), an operator, a right side(cell), and (if not the last row of logic) a conjunction ("AND" or "OR") which ties the rows of logic together. The joined rows make up the rule's logic.

Left Side Logic/Right Side Logic Cells

Each row of logic must have a left side(cell) and a right side(cell) that contain variables and/or values that can be evaluated. For example, both a left cell and a right cell can contain "**XYZ**" (a variable) or "**(TEMP-32)*10**" (a variable and value), while a right cell can also contain "**01/02/92**" (a value).

Operators

The left side logic and right side logic can be evaluated and their results compared to each other using the operator to determine if each row is true or false. For example, "3 xyz ≤ 32.2" uses the operator "≤."

Parentheses

Used to group different rows of logic together into a complex logical expression.

Parent

A variable whose value can enable or disable the data entry value of another variable.

Example: If the response to the parent variable **Arterial blood gas (ABG) done** is false, then the child variable **Intubated when ABG done** cannot accept any data entry value.

Child

A variable whose data entry is enabled or disabled by the value of another variable.

Example: The variable **Intubated when ABG done** can only accept a value when the parent variable **Arterial blood gas (ABG) done** is true.

RULE TYPES AND LOCATIONS

When you design a dictionary, you have the ability to create data entry rules in several different places. The functions are made available where they are most appropriate. For example, after you add variables to a screen, you can select the **Edit Rules** mode and the type of rule you want to create.

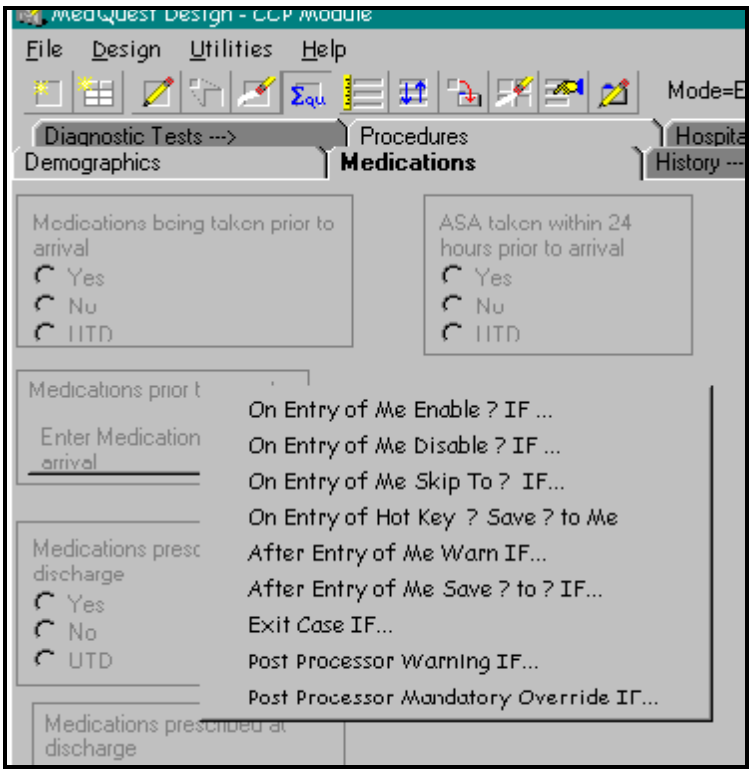
Following is a description of the various design windows where you can add/edit/delete a rule for a variable or a screen:

Rule Type	Design Location			
	Design Window	Order/Indent/ Unindent	Grid Design Window	Screen Design Window
Enable Variable Rule	✓	✓	✓	
Disable Variable Rule	✓	✓	✓	
"Skip To" Variable Rule	✓			
Hot Key Rule	✓			
After Entry Warning Rule	✓		✓	
Save Value to Variable Rule	✓			
Exit Case Rule	✓			
Post Processor Warning Rule	✓			
Post Processor Mandatory Override	✓			
Do Not Load Screen Rule				✓
Disable Screen Rule				✓

The various ways to create rules in these windows are described below:

Access on the Design Window

On the **Design** window, select the Edit Rules option to access the **Edit Rules** mode. When you click on any variable on the screen, select the type of rule to create. You will then be taken into the **Rule Editor**.

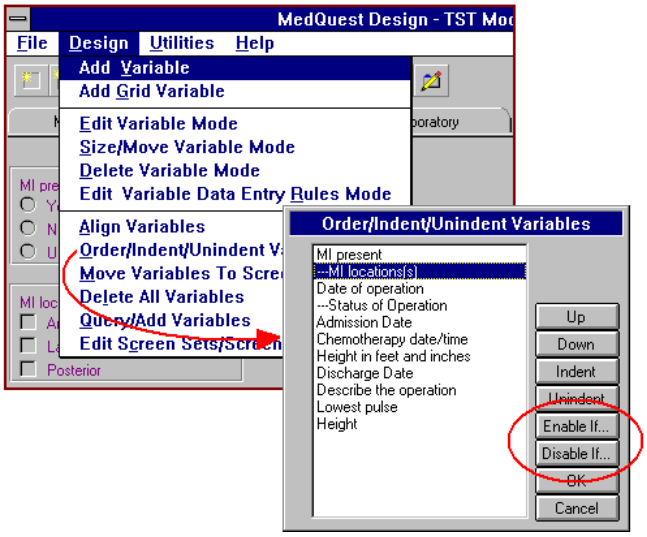


Example 1: To add a warning that a number should be between "0" and "100," select the **Edit Rules** mode and then click on the numeric variable to be validated. From the popup rules box, select "**After Entry of Me Warn IF**" to build the rule.

Example 2: To add a warning that a date should be between the admission and discharge date, select the **Edit Rules** mode and then click on the date variable to be validated. From the popup rules box, select "**After Entry of Me Warn IF**" to build the rule.

Access on Order/Indent/Unindent Variables Window

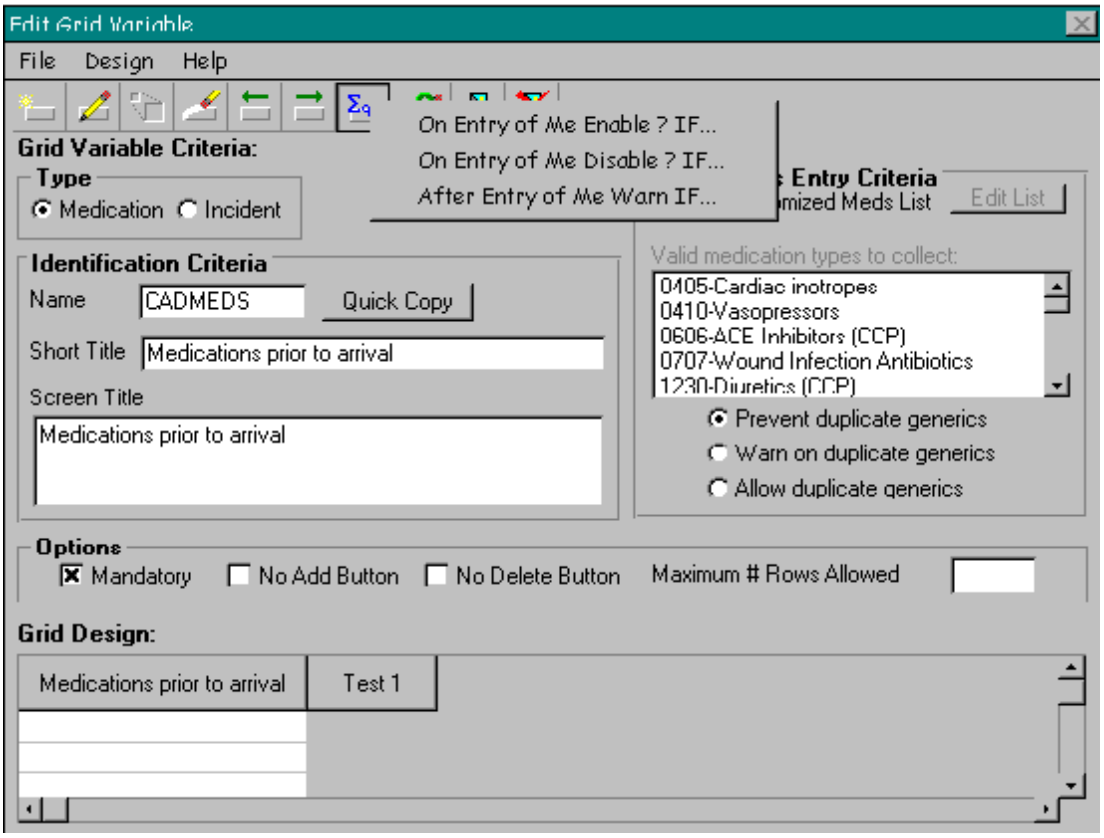
This method of access is recommended for creating data entry enabling/disabling rules (or skip patterns). On the **Design** window, access the **Order/Indent/Unindent Variables** function. To create a parent-child relationship, indent the variable that is to be a child. Select that variable on the list and then select the <ENABLE> or <DISABLE> button to access the **Rule Editor**.



Example: To add a rule to disable (or skip) a variable, select the **Order/Indent/Unindent** function and then select the indented (child) variable to disable. Select the "**Disable If...**" function. The **Rule Editor** will let you build the rule to disable the indented variable when the value of the parent variable changes.

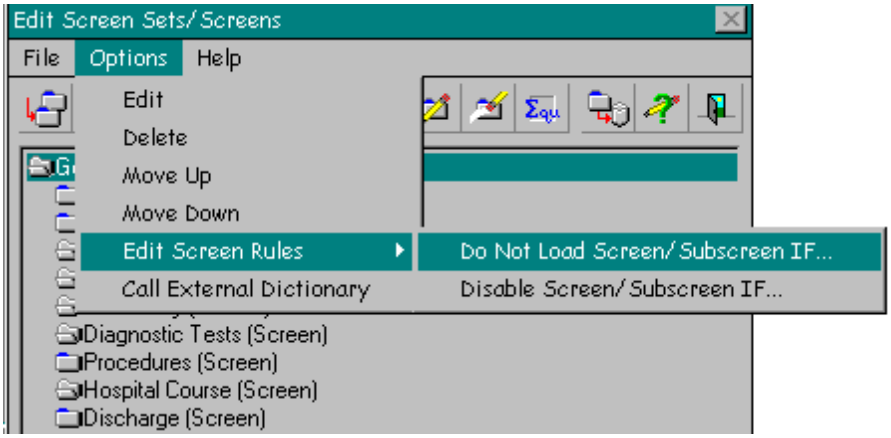
Access on the Edit Grid Variable Window

On the **Edit Grid Variable** window, select the variable in the grid to which you want to add a rule and then select the <EDIT RULES> button. After you select a rule type you will be taken into the **Rule Editor**.



Access on the Screen Design Window

In the **Edit Screens Sets/Screens** window, select the **Edit Rules** icon (or select Edit Rules from the menu) after selecting the screen or subscreen to which you want to add a rule.



Example: To add a screen rule, select the **Edit Screens Sets/Screens** function and then select the screen to be disabled. Select the "**Disable Screen/Subscreen If...**" function. The **Rule Editor** will allow you build the rule to disable the screen when the value of a variable or of another screen changes.

EDIT ENABLE VARIABLE RULE

The Enable **Variable Rule** allows you to specify the logic to enable a variable when the value of another variable is changed (e.g., on entry of one variable, enable another variable if rule is true). This type of rule can be used to create "skip patterns". You can add this rule type to a variable in both the **Design** window and the **Order/Indent/Unindent Variables** window.

This rule type accesses the **Rule Editor** for selecting the variable to be enabled and entering of the logic. You can enter any equation in the **Rule Editor**, but it assumes that you want the parent variable included in the logic (which may not be the case). The parent variable is the variable that you originally clicked on to create the rule. During data entry, the rule executes when the value of the parent variable is entered or changed. If the rule is evaluated as true, the selected child variable will be enabled, otherwise, it will be disabled.

When you select this rule type in the **Order/Indent/Unindent Variables** window, you must have already selected an indented (child) variable from the variable list on the screen. This indented variable will be enabled when the value of the parent variable changes and the logic entered in the **Rule Editor** is true.

Note that you cannot use this rule type with **Sequential Data Entry**.

Rule Editor called from: **Design, Order/Indent/Unindent Variables, Grid Design**

Example: Enable "**MI Location**" if "**MI Present**" is "**Yes**." Move "**MI Location**" beneath "**MI Present**" and indent "**MI Location**." Highlight the "**MI Location**" variable and select the "**Enable If...**" option. Use the **Rule Editor** to enter the logic "**If MI Present is Yes**."

Advanced User: Optimum Programming Approach

It is recommended that you build as many enable and disable rules as possible on the **Order/Indent/Unindent Variables** window. This method can reduce the number of equations you need to create. The enable/disable rules are special rules that utilize the indentation level of a variable to determine whether the variable is enabled or disabled. All child variables (and other descendants) of a disabled parent will automatically be disabled. Based on this logic, the hierarchy created by the use of indentation provides a convenient structure to minimize the logic that you must create.

Note that if you do not specify an equation for an indented variable, it will always be disabled.

EDIT DISABLE VARIABLE RULE

The above discussion for the **Enable Variable Rule** also applies to the **Disable Variable Rule**. The difference is that if the rule is true then the child variable will be disabled; otherwise it will be enabled.

Note that you cannot use this rule type with **Sequential Data Entry**.

Rule Editor called from: **Design, Order/Indent/Unindent Variables, Grid Design**

Example: Disable variable **Last Systolic Blood Pressure** if variable **Discharge Status** is "**Expired**." On the **Design** window, select the **Edit Rules** mode, click on the **Discharge Status** variable and select the "**On Entry of Me Disable ? IF**" rule type. Inside the **Rule Editor**, select the variable to disable (select "**Last Systolic Blood Pressure**") and then enter the logic "**If Discharge Status is Expired**."

Advanced User: Disable/Enable Variables on Other Screens

You can disable/enable variables on other screens when you enter values on a given screen during data entry. For example, if the **Discharge status** (on the **Discharge** screen) is "**Expired**", you can disable the **Lab values at discharge** (on the **Labs** screen).

EDIT "SKIP TO" VARIABLE RULE

The "skip to" **Variable Rule** will skip, or disable, variables between one variable and another if the rule is evaluated as true. To create the rule, select the variable to skip from, the logic of the rule, and the variable to skip to. You may not skip to a variable that appears (in sequence) before the variable from which you are skipping.

Rule Editor called from: **Design**

Advanced User: Using "Skip To" Rules with Enable/Disable Rules

It is recommended that you **do not** mix enable/disable rules with "skip to" rules. This may introduce inconsistent logic. For example, a "skip to" rule that skips variables that have disable rules may cause some variables to be skipped (skip rule is true) which have been enabled (disable logic is false, therefore enabling the variable).

If you mix rule types, it is recommended that you only mix "skip to" rules with enable rules, since a blank value (after a "skip to" rule is executed) would cause the enable rule to be false (unless your rule evaluates to true for a blank value). This result ensures that the expected variable is disabled. Also, ensure that all variables that are enabled by the rule are included in the set of variables between the "skip from" and "skip to" variables.

Advanced User: Using "Skip To" Rules with Sequential Data Entry

In sequential data entry you are assumed to be entering data in a specific order, therefore, rules (other than "skip to") that can enable or disable variables anywhere in the sequence are not permitted. You can use "skip to" rules with **Sequential Data Entry** but you cannot use enable/disable variable rules or disable screen rules.

EDIT HOT KEY RULE

The **Hot Key Rule** allows you to specify for a given hot key a value or variable to save to the variable being entered. When you create the rule, you are allowed to specify only the hot key and the value or variable to save. Note that you cannot create a hot key rule on a grid type variable.

Rule Editor called from: **Design**

Example: Copy the **Admission Date** to a **Lab Test Date** variable by pressing the <A> key.

EDIT AFTER ENTRY WARNING RULE

This rule type is used to issue a warning that appears after completing entry of a variable. For example, this rule may be used to warn the user that a value entered should be between an upper and lower bound.

To add this type of rule, select the variable for which you want to add an edit warning, then select the rule type. When you select the variable, you will be taken directly into the **Rule Editor** to enter the logic. In data entry, the rule is executed whenever you finish entry of a variable (i.e., leave the variable). If the conditions of the rule are true, a warning message (the English translation of the rule) is displayed.

Rule Editor called from: **Design, Grid Design**

Example: Issue a warning if the date a medication is administered is not between the admission and discharge dates.

EDIT SAVE VALUE TO VARIABLE RULE

This rule type allows you to save a value to a selected variable after you have completed entry of another variable.

Rule Editor called from: **Design**

Example: Save a value of "**0**" to the variable **Lowest Pulse** if the variable **Discharge Status** is "**Expired**."

After you have pointed to the variable whose entry invokes the rule and selected this rule type, you can select the variable to save to, the value to be saved, and enter the logic for the rule using the **Rule Editor**. In data entry, the rule executes when you finish entering a value (i.e., you leave the variable).

Advanced User: Save a Value to a Variable That Is Disabled

It is possible for you to have this rule save a value to a variable that is disabled through the Enable/Disable rules. MedQuest does not check for this variable's status since it assumes that, as a programmer, you will account for these conditions with the logic you create. MedQuest always saves the value if the rule executes to true.

EDIT EXIT CASE RULE

This rule type forces the data abstractor to stop data entry and exit the case. To add this rule, select a variable to include in the logic and then select the rule type.

Rule Editor called from: **Design**

Example: Stop entry of a case if the variable **Discharge Status** is "**Expired**."

All exit rules are always executed when a case is first opened and after data for each variable is entered. If any exit conditions exist, the abstractor will not be allowed to continue data entry. A warning message is displayed on each action after an exit case rule is evaluated as true.

Advanced User: Too Many Exit Rules

If there are too many exit rules, there will be a noticeable decrease in the speed of the data entry system since all of the exit rules execute after entry of each variable. Therefore, avoid adding too many exit rules to the system.

EDIT POST PROCESSOR MANDATORY OVERRIDE RULE

Select this rule type to create logic that will cause the system to override a **Post Processor Warning** message displayed when a mandatory variable has not been entered.

To add this rule type, select the variable to apply the post processor mandatory override and then select the rule type. In the **Rule Editor,** you can enter the logic associated with overriding the warning message.

Rule Editor called from: **Design**

EDIT POST PROCESSOR WARNING RULE

Select this rule type to display a general warning message in the **Post Processor Warning** window (after entry of a screen or after entry of a case).

To add this rule type, select the variable to apply the post processor warning and then select the rule type. In the **Rule Editor** you can enter the logic associated with the warning and also enter a customized warning message to appear in the **Post Processor Warning** box.

Rule Editor called from: **Design**

Example: Warn the user to check the value of the variable **High Blood Pressure** because the value of the variable **Hypertension** is "**Yes**."

Advanced User: Cascade Rules

Since you can use MedQuest to create a set of customized complex rules, it is possible that you, as the programmer, may introduce logic bugs into your data entry system in the following manner: too many rules that reference the same variables in different ways can lead to cascading logic that can put the rules' results into an unknown state. To avoid cascading, MedQuest will only execute the rules associated with a given variable. The last rule specified in the **Rule Editor** takes precedence.

EDIT DO NOT LOAD SCREEN RULE

To prevent a user from entering data on a screen until certain conditions are met use the **Do Not Load Screen Rule**. This rule can be added to any screen, subscreen or screen call.

Typically, this rule is used to force the user to enter data on screens in a certain order or to ensure that data will not be entered on other screens until certain key screens are filled in first. This type of rule also enables you to include logic to test if a screen is empty (no data is entered) or filled (all mandatory variables have a value). Note, that you cannot add this rule type with **Sequential Data Entry**.

Rule Editor called from: **Screen Design**

EDIT DISABLE SCREEN RULE

Select the screen/subscreen to be disabled and then enter the logic. The variables on the screen are disabled only if the rule is evaluated as true; otherwise, no action takes place. By default, the variables on a screen are enabled when that screen is loaded.

This rule will accept any logic, including whether a screen is empty (no data entered on the screen) or filled (all mandatory variables have a value). Note that you cannot add this rule type with **Sequential Data Entry**.

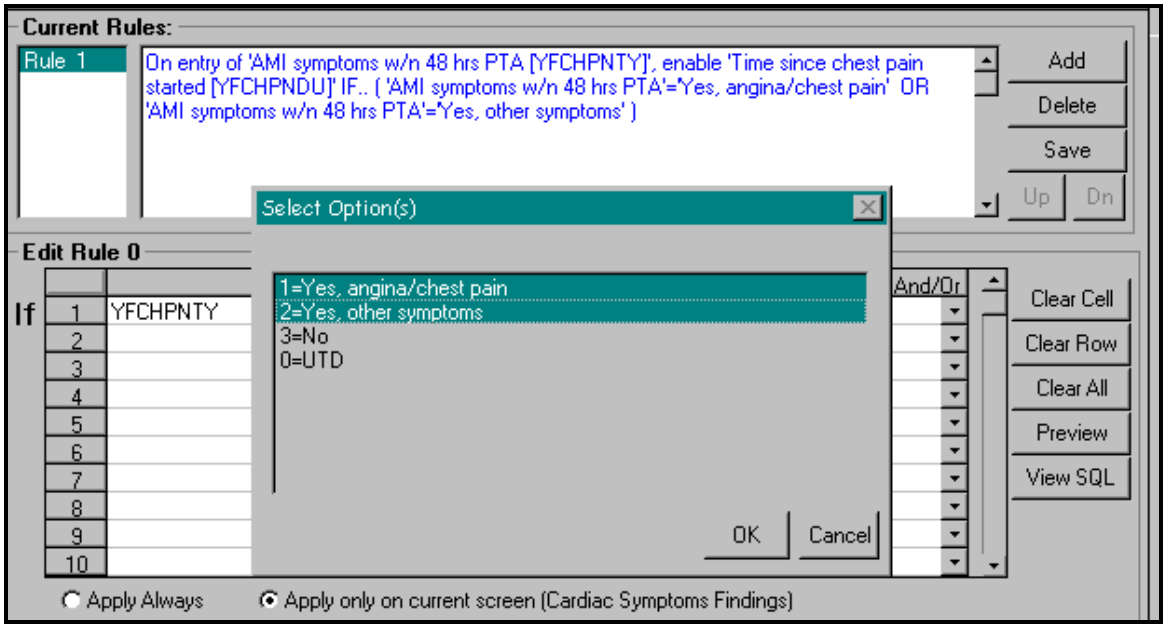
Rule Editor called from: **Screen Design**

Advanced User: Do Not Load Screen Rules Versus Disable Screen Rules

The **Do Not Load Screen Rule does not** disable that screen. It simply prevents a screen from being loaded under certain conditions. If there are mandatory variables on a screen that cannot be loaded, the abstractor will not be able to complete the case in data entry since post processor errors will exist. To **disable** the variables on a screen for a set of logic, use the **Disable Screen Rule**.

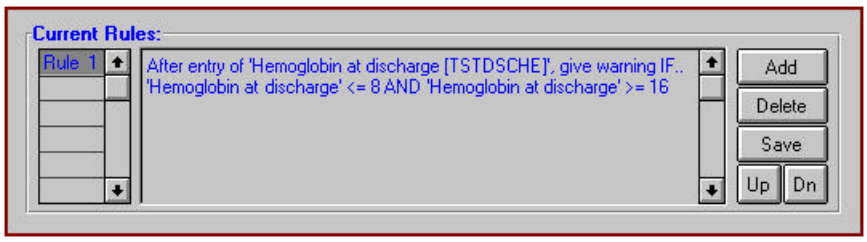
RULE DEVELOPMENT

RULE EDITOR WINDOW



The **Rule Editor** window, entitled "Edit Rules", is divided into two parts: Current Rules and Edit Rule. The **Current Rules** section displays all the rules that have been created for the selected variable. The **Edit Rule** section is a grid that displays the rule currently being created or edited. The window provides several assistive devices for creating data entry rules.

CURRENT RULES



When the **Rule Editor** opens the **Edit Rules** window, the existing rules are listed at the top of the window. The current rule is highlighted and its English translation is displayed in the box at the top of the screen. To view the English translation, select the rule from the list in the upper left corner of the screen.

The logic for the current rule will be displayed in the grid in the middle of the screen where it can be edited.

ADD A RULE

To add a rule, click on the <ADD> button. This will clear the logic from the previous rule and prompt you to save it. The system will pre-load the grid with the "expected" logic for the new rule. The expected logic can be changed, and is simply the **Rule Editor's** best guess as to the logic you might want for the type of rule to be added. For example, for a disable rule, the variable name for the "parent" variable will be placed in the left cell, and the operator will be changed to "=." This is because, in most circumstances, you will be disabling the "child" variable if the "parent" is equal to some value. Different types of variables populate the grid logic in different ways.

DELETE A RULE

To delete a rule, select the <DELETE> button. If you delete a rule it will be removed from the data dictionary and you will not be able to recover it.

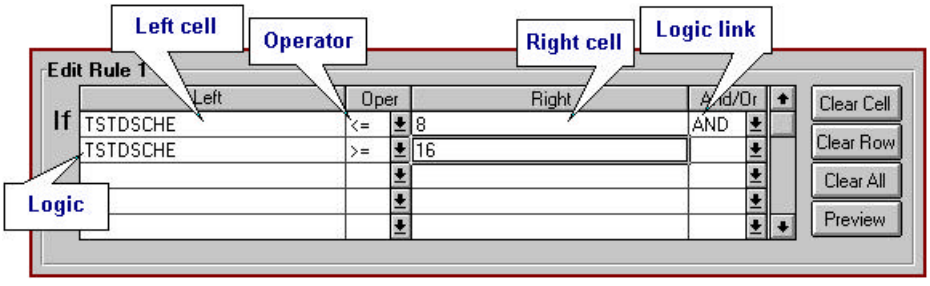
SAVE A RULE

You have the option to save a rule as you are designing it to ensure that your work does not get lost if you accidentally power down, etc. In addition to being able to explicitly save a rule, the **Rule Editor** will ask you whether you want to save a rule whenever an action changes the current rule on which you are working.

ORDER RULES

You can change the order in which the rules listed in the **Edit Rules** window are executed. Highlight a rule and click on the <UP> or <DOWN> button. **The last rule in the order is the last one executed and takes precedence over the actions of any previous rules.**

EDIT RULE GRID

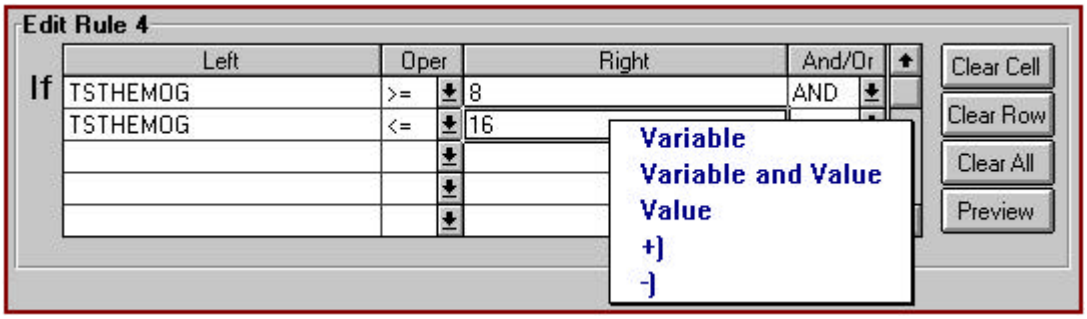


Most rule construction takes place in the **Edit Rule** grid. To add to or edit a cell, click on that cell. The **Rule Editor** requires entry of rules from left to right. First enter the expression for the left cell, then the right cell.

When you click on a left cell, a popup menu will appear with the following options: enter a variable, enter a variable and a value, add or remove a left parentheses. You cannot enter only a value into the left cell. This popup menu is context sensitive - the options that appear depend on the MedQuest variable type.

Select the operator in the operator cell by clicking on the scroll bar. This will popup the available operators.

When you click on the right cell, a popup menu will appear asking if you want to enter a variable, enter a variable and a value, enter a value, add a screen, add the key word "EMPTY" or "FILLED," or add or remove a right parentheses. The set of options presented is context sensitive depending on the type of variable entered in the left cell.



VARIABLE. Select the variable from the variable selection screen.

VALUE. Enter a value or values specific to the type of variable in the left cell. For example, if the variable in the left cell is an Option (pick one) you will be presented with the options for that variable when you select the value menu. For certain variable types, you will have the option to select "UTD" or "Any Valid Value" (for that variable type).

VARIABLE PLUS VALUE (EQUATION). When you select to add a variable plus a value, the **Equation Design** window will appear. The **Equation Design** window enables you to build a more complex set of logic for certain variables. Following are the types of equations that can be built for the different variable types:

- **Date**. Variable + or - days (e.g., CDISDAT-2 days)
- **Time**. Variable + or - minutes (e.g., CTIME+20 minutes)
- **Date/Time**. Variable + or - hours (e.g., CDTIME+48 hours)
- **Number (no units)**. Complex equation
- **Number (units)**. Complex equation
- **Number (set)**. Complex equation

SCREEN. This is available only for rule types **Do Not Load Screen If** and **Disable Screen If**. To be included in the logic, the screen must be selected in the left cell.

EMPTY. This is available only in the right cell (when a screen is selected in the left cell) to define logic for an empty screen (no variables with values on the screen). Note that "Empty" **is not** the opposite of "Filled."

FILLED. This is available only in the right cell (when a screen is selected in the left cell) to define logic for a filled screen. "Filled" is defined as all mandatory variables having a value. Use the "<>" operator to define the opposite of filled.

CREATE A COMPLEX EQUATION

If you are building a complex equation, you can build the string by using the assistive devices to add a variable, add operators and parentheses, and enter numeric values. When you work with a date or a time variable you are limited to adding or subtracting the applicable time unit.

Example: To create ((CWEIGHT*22)/3), select the variable **CWEIGHT**, select the operator "*****" and enter "**22**" in the right cell. Click on parenthesis "**+(**" and "**)+**," click on operator "**/**" and enter "**3**". If you prefer, you can type the equation in the box.

USE OTHER EDIT RULE GRID FUNCTIONS

You can use the other assistive devices in the **Rule Editor** to clear a cell you have highlighted, clear a row of logic, clear all the logic, or preview the English translation of the logic being edited. You may also be required to enter a variable and a value to save, which is dependent on the type of rule selected.

RULE EXECUTION IN DATA ENTRY

RULE EXECUTION POINTS IN DATA ENTRY

The rules that you create are executed on different events during data entry. Below is a list of events that cause rule execution. Note that some rules may be executed more than once for different purposes. For example, it is useful to check if there is any logic that should stop data entry on entering a case, when loading a screen, and after entering a value.

Rule Type	Execution Point							
	On Loading Case	On Loading Screen	On Entry of Variable	On Entry of Grid	After Leaving Case	After Leaving Screen	After Entry of Variable	After Entry of Grid
Enable Variable Rule		✓	✓	✓	✓	✓		✓
Disable Variable Rule		✓	✓	✓	✓	✓		✓
“Skip To” Rule		✓	✓		✓	✓		
Hot Key Rule			✓					
After Entry Warning Rule							✓	✓
Save Value to Variable Rule							✓	
Exit Case Rule	✓	✓					✓	
Post Processor Warning Rule					✓	✓		✓
Post Processor Mandatory Override					✓	✓		
Do Not Load Screen Rule		✓						

Disable Screen Rule	✓	✓			✓			

During data entry, all data entry rules that have been built into the dictionary will be executed. Each rule is executed on the occurrence of events that take place during data entry.

EXECUTE ON LOADING A CASE

EXIT CASE RULE. The exit case rule is always executed when loading a case. This ensures that you return to the same logical state each time you open that case.

COLOR TAB GREEN, FILLED SCREEN. If you have selected this option in design preferences, all of the enable, disable, "skip to" and disable screen rules will execute for all of the screens. Note that this option may cause a noticeable decrease in processing speed.

COLOR TAB BLUE, PARTIALLY FILLED SCREEN. If you have selected this option in design preferences, all of the enable, disable, "skip to" and disable screen rules will execute for all of the screens. Note that this option may cause a noticeable decrease in processing speed.

DISABLE SCREEN RULE. The disable screen rules execute first. If the logic is true then the variables on the screen are disabled.

EXECUTE ON LOADING A SCREEN

ENABLE VARIABLE RULE. If the result of the rule logic is true, the variable is enabled; otherwise it is disabled. If there is no equation for an indented variable, it will always be disabled.

DISABLE VARIABLE RULE. If the result of the rule logic is true, the variable on the screen is disabled; otherwise it is enabled. If there is no equation for an indented variable, it will always be disabled.

"SKIP TO" VARIABLE RULE. If the result of the rule logic is true, the variables between the one being entered and skipped to are disabled; otherwise they are enabled.

EXIT CASE RULE. If designated values or a combination of values is true, you are warned to exit the case.

Note: The Exit Case rule does not prevent you from changing values, it just warns you that you should exit the case. This allows you to correct a value that you may have entered incorrectly that forced the case to exit.

Advanced User: Execution Order of Disable Screen, Enable, Disable and "Skip To" Rules

On loading a screen, all disable screen rules are executed first. If a screen is disabled, no rules that will enable or disable a variable on that screen are executed. The enable, disable, and "skip to" rules are executed for each variable in the order of the variables on the screen. The rules themselves are executed in the order in which they were entered, unless that order has been changed. You can change the executing order (for each variable) of each rule type (if you have multiple rules) by using the <UP> or <DOWN> keys in the **Rule Editor**.

You **cannot** change the order of execution of these rules relative to each other. (For example, you cannot make all the "skip to" rules execute first, followed by the enable rules.) The execution order is dependent on the order in which the rules were created in the database. For this reason it is recommended that you do not mix enable, disable and "skip to" rules on or across the same variables.

DO NOT LOAD SCREEN RULE. If the result of the rule logic is true, the screen(s) is not loaded.

DISABLE SCREEN RULE. The disable screen rules are executed first. If the logic is true, the variables on the screen are disabled.

EXECUTE ON ENTRY OF A VARIABLE

ENABLE VARIABLE RULE. If the result of the rule logic is true, the variable is enabled; otherwise it is disabled.

DISABLE VARIABLE RULE. If the result of the rule logic is true, the variable on the screen is disabled; otherwise it is enabled. If a parent variable is disabled, all of its descendants will be disabled.

"SKIP TO" VARIABLE RULE. If the result of the rule logic is true, the variables between those being entered and skipped to are disabled; otherwise they are enabled.

HOT KEY RULE. This rule is executed (to save a value from another variable to the one being entered) when the assigned hot key is pressed.

EXECUTE ON ENTRY OF A GRID

ENABLE VARIABLE RULE. If the result of the rule logic is true, the variable is enabled; otherwise it is disabled.

DISABLE VARIABLE RULE. If the result of the rule logic is true, the variable on the grid is disabled; otherwise it is enabled. If a parent variable is disabled, all of its descendants will be disabled.

EXECUTE AFTER LEAVING A CASE (WITH POST PROCESSING AFTER CASE OPTION)

All disable screen, enable, disable and "skip to" rules for all screens are executed for this event.

MANDATORY RULE. If the variable is mandatory, you will be warned if that variable is enabled (you had a chance to enter it) and does not have a value entered. You will be required to enter data or make changes before being able to complete the case.

POST PROCESSOR WARNING RULE. If there is any post processor warning rule that evaluates to true, you will be presented with a warning message. You can choose to ignore it.

EXECUTE AFTER LEAVING A SCREEN (WITH POST PROCESSING AFTER SCREEN OPTION)

All disable screen, enable, disable and "skip to" rules for the given screen are executed for this event.

MANDATORY RULE. If the variable is mandatory, you will be warned (if you have post processing after screen turned on) if that variable is enabled (you had a chance to enter it) and it does not have a value entered. You will be required to enter data or make changes before being able to complete the case, unless the variable has a post processor mandatory override rule whose conditions are met.

POST PROCESSOR WARNING RULE. If there is any post processor warning rule that evaluates to true, you will be presented with a warning message. You can ignore it and still be allowed to complete a case.

COLOR TAB GREEN, FILLED SCREEN. If you have this option selected, all of the enable, disable, "skip to" and disable screen rules will execute for all of the screens. Note that this option may cause a noticeable decrease in processing speed.

COLOR TAB BLUE, PARTIALLY FILLED SCREEN. If you have this option turned on, all of the enable, disable, "skip to" and disable screen rules will execute for all of the screens. This option is only exercised for the screen on which you were entering data. Note that this option may cause a noticeable decrease in processing speed.

EXECUTE AFTER ENTRY OF A VARIABLE

AFTER ENTRY WARNING RULE. If the warning rule is true, a warning message will be displayed. A common type of warning is a variable outside of a specified range (e.g., temperature > 115 degrees). You can override these warnings by selecting the <OK> button, or you can select the <RETRY> button to make changes to the variable. If the **Retry Only** option was selected in the **Design**, you will always be forced to enter an acceptable value.

SAVE VALUE TO VARIABLE RULE. After entry of a variable, if the rule is true, the specified value will be saved to the specified variable. If you overwrite the value saved by the rule, your entry will be the one saved in the database unless the rule is executed again.

Example: Program the logic to save "**0**" to the variable **Lowest systolic blood pressure** if the patient has "**Expired**."

EXIT CASE RULE. If the exit case rule is true, you are warned to exit the case.

Note: The Exit Case rule does not prevent you from changing values, it just warns you that you should exit the case. This allows you to correct a value that forced the case exit that may have been entered incorrectly.

EXECUTE AFTER ENTRY OF A GRID (IF YOU HAVE POST PROCESSING AFTER GRID TURNED ON)

MANDATORY RULE. Any grid variable that is mandatory and does not have a value entered will have an error warning displayed.

DEBUGGING RULES

It is often advantageous in complex systems to turn on the Debug On option in **Preferences**. This will display a floating popup window during data entry. The **Debug** window will display a set of detailed information you will find useful as your rules execute. First, each rule that executes (and the order that rule was executed) will be displayed in a list in the window. Next, for each rule executed, you can display the details about that rule (raw logic, English translation of the rule, result of rule (true or false) and values of the tokens (variables, values, etc.)) that comprise the rule. This window is not accessible during grid data entry.

APPENDIX A: RULES FOR BUILDING RULE LOGIC

Following are the rules you must follow in the construction of the logic for a rule.

- Each row of the **Edit Rule** grid must contain a left cell, an operator, a right cell, and a logic link "AND" or "OR" that ties the rows together for all but the last row.
- The left cell must be entered before the right cell is entered.
- One variable, variable + value or screen, can be entered in the left cell.
- One variable, variable + value, value, or key word "EMPTY" or "FILLED" can be entered in the right cell.
- The variable in the left cell determines the edit type required for the entry in the right cell. For example, if the variable in the left cell is a number, the entry in the right cell must also be a number.
- Any variable entered must be present in the master table of the module dictionary.
- You can add as many open parentheses "(" to the left cell as you want and as many closed parentheses ")" to the right cell as are necessary.

APPENDIX B: ADVANCED USERS– ENABLING/DISABLING/SKIP TO EQUATIONS

The enable/disable rules are sufficiently complex in their features that some discussion of their function is necessary. This discussion is presented below.

ENABLING/DISABLING FROM THE ORDER/INDENT/ UNINDENT DESIGN WINDOW

If you are enabling/disabling variables on the same screen as the variable that invokes the enabling/disabling action, it is recommended that you use the functions provided on the order/indent/unindent design window. By indenting variables into what are called parent/child relations, you can make more effective use of the equations.

For example, to enable the following variables:

Variable	If yes, enable variables 2,3	
----------	Variable 2	
----------	Variable 3	If yes, enable variables 4,5
----------	----------	Variable 4
----------	----------	Variable 5

On the order/indent/unindent design window, indent variables 2 and 3 one level down and indent variables 4 and 5 two levels down. Then highlight each child variable (2,3,4,5) and add an equation to enable each variable based on the parent variable's value. This results in four equations.

The hierarchy that the indentation establishes provides you with an additional capability that you do not have to program. This rule is:

"Any indented variable will be disabled if its parent is disabled."

This makes it possible to quickly build a hierarchy of variables and rules when such a relationship has been established.

Please note that this functionality provides you with the same abilities to enable/disable equations that were provided in older versions of MedQuest.

Executing Enable/Disable Equations on All Screens or a Specified Screen

When you build an enable/disable rule, you are given the option of selecting whether you want the equations executed only for the screen containing the variables or for all screens. This functionality is provided for several reasons:

- Using the same variable on different screens or different screen sets when that variable has different rules associated with it. For example, you may have the "History of MI" variable enable the "Time of MI" and "Date of MI" on one screen, and to enable just the "Date of MI" on another screen. Be careful with this type of functionality - it is easy to create relationships that could lead to unexpected results.
- Executing enable/disable equation all of the time to enable or disable a variable on another screen. When you do this, the enable/disable rule will execute in the following ways:
  - When the variable that executed the equation is changed
  - When a screen is loaded that contains the variable being enabled/disabled

If you do not elect to execute the equation all the time, the rule will only execute when the variable on which the equation is based is changed. Differing results may occur depending on the order in which the data are entered on the screen.

When you build enable/disable rules in the order/indent/unindent design window, **the default is to execute the rules only on that screen**. Therefore you should build enable/disable equations for variables that are in parent/child relationships to each other on the same screen.

When you build enable/disable rules for variables that are on different screens (using the Equation Mode option from the main MedQuest Design window) **the default is to execute the rules all the time**.

ENABLING/DISABLING VARIABLES ON DIFFERENT SCREENS

For the value of a variable on one screen to enable/disable a variable on a different screen, use (as discussed above) the Equation Mode option from the main MedQuest design window and **be certain to select the option to execute the rule all the time**. This will insure that the equation will run if you select data entry and first open the screen with the variable to be enabled/disabled.

Sound informatics engineering practice dictates keeping variables in a hierarchical relationship (parent/child) together on the same screen. However, if there are too many variables to accomplish this, you can still use the features provided on the order/indent/unindent design window to help you. For example, you could do the following:

Screen 1

Variable 1	If yes, enable variables 1,2 (this screen)	
----------	Variable 2	
----------	Variable 3	If yes, enable variable 4 (all screens)

Screen 2

Variable 4	If yes, enable variables 5,6 (this screen)	
----------	Variable 5	
----------	Variable 6	

This logic takes advantage of the fact that if a variable is disabled, all of its descendants will be automatically disabled.

There is an alternative to performing this function that might be useful. The following demonstrates the use of the "Save Value to" rule to perform an action similar (but not exactly the same) as the functionality described above:

Screen 1

Variable 1	If yes, enable variables 1,2	
----------	Variable 2	
----------	Variable 3	If yes, save yes to variable 4 on Screen 2

Screen 2

Variable 4	If yes, enable variables 5,6
----------	Variable 5
----------	Variable 6

Note that with this approach you are introducing a different functionality so that Variable 4 would never be disabled and could always be modified. Sometimes you will find that the relationship of variables to each other is not hierarchical but involves an action to one variable based on the value entered for another.

WHEN TO USE ENABLE, WHEN TO USE DISABLE

Use the enable rules when you want a blank (empty) value to disable a variable. (Null values in equations will cause logic to be false unless you explicitly program otherwise.) You should use the disable rules if you want a blank (empty) value to enable a variable.

It is recommended that you avoid mixing enable rules with disable rules since both rules perform in a similar way but provide opposite results. Mixing enable and disable logic can make the task of debugging the logic more difficult.

You can always modify the rule logic to change an enable rule to a disable rule (or the other way around). Before you begin designing a module, it is recommended that you choose which way you want to enable/disable variables and adhere to one strategy.

WHAT HAPPENS WHEN ENABLE / DISABLE RULES EXECUTE

When an enable rule executed, if it is true, the action fieldname (variable) is enabled; **otherwise it is disabled.**

When a disable rule executes, if it is true, the action fieldname (variable) is disabled; otherwise it is enabled.

When a variable is disabled, its value is automatically cleared whether on the same screen or a different screen than the invoking variable. The system can issue a warning when disabled variables contain data which is about to be cleared. This option can be selected from the menu item “Preferences” during design.

RELATIONSHIP OF ENABLE / DISABLE RULES TO POST PROCESSOR, MANDATORY

A variable designated as mandatory, if disabled on any screen, will not need to be entered to mark a case as complete. Please note that it is possible for you to introduce your own logic bugs by using a variable on more than one screen and using different enable/disable equations for that variable. For example, you might have the following situation:

Screen 1

Variable 1 If yes, enable variable 2

---------- Variable 2

Screen 2

Variable 3 If yes, enable variable 2

---------- Variable 2

This might result in variable 2 being enabled whether you open screen 1 first or screen 2 first (depending on the values for variables 1 and 3). As such, the results shown in the post processor would also vary. The post processor can only deal with one instance of variable 2 and only understands there to be one enable/disable status for that variable. When the post processor executes, it cycles through each screen (by order) and each variable on that screen (by order) and executes all of the equations for any variable that could be enabled/disabled by a rule on that screen or all screens.

To avoid conflicting relationships between variables and logic, avoid constructs such as the one above or complete logic in the rules that will make the relationships exclusive and without conflict.

CONFLICTING ENABLE / DISABLE LOGIC

As described above, you may introduce logic that could lead to a programming bug. For example, you might have the following situation:

Screen 1

Variable 1 If yes, enable variable 4 (all screens)

Variable 2 If no, enable variable 4 (all screens)2

Screen 2

Variable 4

This imposes two problems. First, when you enter variable 1 or variable 2 on Screen 1, the order of entry will influence whether variable 4 is enabled or disabled.

Second, when you open Screen 2, since both rules execute, the second rule will take precedence in the logic. For example, you could have variable 1 = yes and variable 2 = yes which would result in variable 4 being disabled. This would occur because the rule for variable 1 would execute first (enabling variable 4) and then the rule for variable 2 would execute (disabling variable 4 since it did not enable variable 4).

If a logical bug is introduced, it will need to be fixed by insuring that you have "logical integrity." This is accomplished by trying to make the rules all inclusive. For example:

Screen 1

Variable 1 — If yes, or variable 2=no, then enable variable 4

Variable 2 — If no, or variable 1=yes, then enable variable 4 (Note that this is duplicative logic and you do not have to include it.)

Screen 2

Variable 4

Specifying the relationships between *all* the variables completes the logic.

TOO MANY ENABLE / DISABLE RULES

It is possible to create a data entry system that has too many rules. This may occur if the information being modeled is hierarchically complex (many parent/child relations, many screens, many variables per screen) and logically complex (many logical relationships between variables between screens, logical relationships between more than two variables, complex logic) the data entry system may have an excessive number of rules. This can lead to two problems: performance degradation, and unexpected results due to logical errors that have been introduced.

As you build data entry systems you will discover that there are alternative approaches to different models. Review the contents of this technical document to see if you can model both the variables (information) and the logic (enable/disable rules) in a way that meets your objectives. There are many alternatives presented here and you will also be creating some of your own. Be aware that the more rules you create, the more likely you will have mixed results because of programming bugs that may be introduced. As MedQuest is enhanced, the alternative paradigms for modeling information and logic will be expanded.

"SKIP TO" RULES

The "skip to" rule will disable a variable(s) between two variables (skip from, skip to) if the logic is true, otherwise it will enable that variable(s). It is not advisable to mix enable/disable and "skip to" rules in the same module. If you do mix these rule types, you should not have enable/disable rules that "cross boundaries" of the variables you are skipping from and to.

Following is an example of mixing a "skip to" rule with an enable rule:

Variable 1 — If yes, skip to variable 5

Variable 2

Variable 3 — If yes, enable variables 4,5,6,7

Variable 4

Variable 5

Variable 6

Variable 7

Variable 8

In this case, the hierarchies of the variables (skip 2 through 4 for “skip to” rule and enable 4 through 7 for disable parent/child hierarchy) are mixed. Although it may produce the anticipated results for this example, it will be more difficult to debug your logic. The better solution is to change the rule on variable 3 to "If No or UTD, skip to 8."

Mixing hierarchies is best accomplished by embedding the parent/child hierarchy completely within one skip boundary. For example:

Variable 1 If yes, skip to variable 6

Variable 2 If yes, enable variables 3,4

Variable 3

Variable 4

Variable 5

Variable 6

Please note that the enable rule is preferable in such a construct because a blank value for variable 2 will disable variables 3 and 4 (consistent with the skip rule). However, a disable rule will enable variables 3 and 4 for a blank value.

DISABLE SCREEN RULE

The disable screen rule will completely disable all of the variables on a screen. You should not try to disable all the variables on the screen on which you are working since this is recursive logic (disable all variables on current screen including the variable that caused the current screen to be disabled).

Instead, MedQuest provides logic to disable a screen depending on the status of another screen. A typical rule would be: Disable *Screen 2* if *Screen 1* is not filled. By using the disable screen rule, if you fill in *Screen 1,* enter some variables on *Screen 2*, then return to *Screen 1* and delete a variable's value, all of the values on *Screen 2* will be deleted (on disable). However, it is more efficient if you are trying to control the order in which screens are entered to use the "Do Not Load Screen If" rule.

APPENDIX C: TUTORIAL— WORKING WITH RULE EDITOR

INTRODUCTION

The ***MedQuest Design Tutorial 3: Working With Rule Editors*** is the third in a series of tutorials prepared to help you gain an understanding on how to use the MedQuest **Rule Editor** function to design rules that will execute during data entry. This tutorial outlines the basic concepts necessary for you to begin creating your own data entry rules. It is recommended that you adhere closely to the instructions provided; they will enable you to explore the major design activities necessary for creating those items.

This tutorial assumes that you are already familiar with how to create a module, add screens to a screen set, and add variables.

TUTORIAL STRUCTURE

The ***Tutorial*** is divided into topics. Each topic contains, where applicable:

- **Explanation**. Lists the functions provided for the topic being described as well as the definition and/or the explanation regarding those functions. This provides you with an understanding of how the tool works and is **not** to be used for testing the tool.
- **Exercise(s)**. Provides examples of activities that you can perform to practice using the tool.

CREATE VARIABLES FOR THE EXERCISE

If you have already created a test module (TST) during the **Tutorial 1** exercises, you can use those variables for this exercise. If you have not yet created a test module during the **Tutorial 1** exercise, create the TST module as an empty module and add the following screens and variables:

- Screen *Laboratory* (Name LABS)
  - TSTHEMOG (Short title and Screen title: Hemoglobin at discharge) - Variable type Number
  - TSTBLDCU (Short title and Screen title: First blood culture collected) - Variable type Option (Pick one) with "Yes," "No," and "UTD" as options
  - TSTBLDDT (Short title: Date blood culture collected and Screen title: Date collected) - Variable type Date
  - TSTBLDT (Short title: Time blood culture collected and Screen title: Time collected) - Variable type Time
- Screen *Medications*(Name MEDIC)
  - TSTMDARV (Short title and Screen title: Meds given after arrival) - Variable type Option (Pick one) with "Yes," "No," and "UTD" as options

- TSTMDNM (Short title: Meds given after arrival name and Screen title: Names of medications) - Variable type Medication (Create as a grid variable). Then add two related variables on the grid
  - TSTMDDT (Short title and Screen title: Date Medication given) - Variable type Date
  - TSTMDRT (Short title and Screen title: Route was IV) - Variable type Option (Pick one) with "Yes," "No," and "UTD" as options

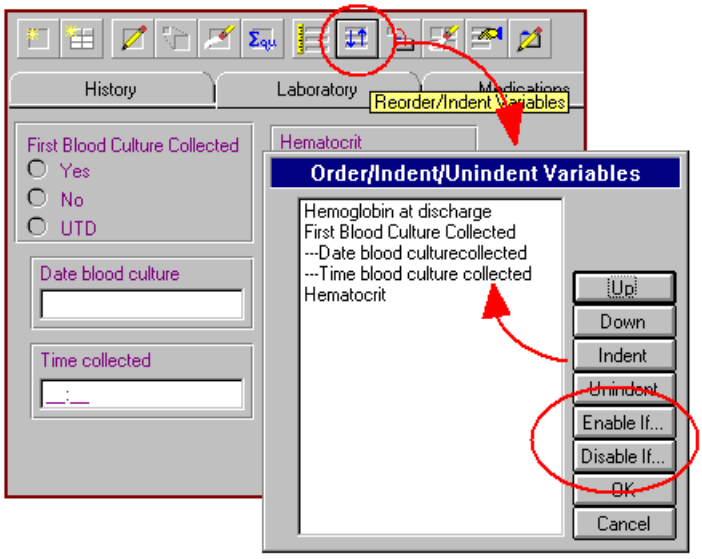
- Screen *Procedures* (Name PROCD)
  - TSTADMDT (Short title and Screen title: Admission Date) - Variable type Date
  - TSTDSCDT (Short title and Screen title: Discharge Date) - Variable type Date
  - TSTPULSE (Short title and Screen title: Lowest pulse) - Variable type Number
  - TSTMI (Short title and Screen title: MI present) - Variable type Option (Pick one) with "Yes," "No," and "UTD" as options
  - TSTMILOC (Short title and Screen title: MI locations(s)) - Variable type Option (Pick one or more) with Anterior (septal) (ant.), Lateral (lat.), Posterior (post.), Inferior (inf.), Subendocardial (non Q-wave), Other (e.g., basal, Apical), and "UTD" as options
  - TSTOPST (Short title and Screen title: Status of Operation) - Variable type String

- Screen *Discharge* (Name DISCHARGE)
  - TSTDSCST (Short title and Screen title: Discharge Status) - Variable type Option (Pick one) with "Discharged," "Expired," and "UTD" as options

The following exercises are designed to familiarize you with building data entry rules using the MedQuest **Rule Editor** function.

DATA ENTRY RULES FOR BEGINNERS

Enable a Child Variable



On the **Laboratory** screen, enable variable **Date collected** if the answer to variable **First blood culture collected** is "**Yes**."

Step 1: Click on the <ORDER/INDENT/UNINDENT VARIABLES> button. Highlight variable **Date blood culture collected** and indent it by clicking on the <INDENT> button. Indent variable **Time blood culture collected**.

Step 2: Highlight variable **Date blood culture collected** and click on the <ENABLE IF...> button. Since this is your first rule, you will automatically be in the **Add Rule** mode. On the **Edit Rules** window, click in the right column of the first row of the grid in the **Edit Rule** box and click on the **Value** option on the popup box.

Step 3: Highlight option "**YES**" on the Select Options window and click on the <OK> button. The value to save will appear in the right column.

Step 4: Click on the <SAVE> button in the *Current Rules* box. The English translation you just created is displayed in the *Current Rules* box.

Follow the steps described above to enable the variable **Time blood culture collected** if the answer to variable **First blood culture collected** is "**Yes**."

Disable a Child Variable

On the **Procedures** screen, disable variable **MI location(s)** if the answer to **MI present** (**TSTMI**) is "**No**" or "**UTD**."

The steps to create this rule are the same as the ones for creating an enable rule, except that you will select the <DISABLE IF...> button instead of <ENABLE IF...> from inside the **Order/Indent/Unindent** window.

Warn on Entry of the Variable

Set the rule to warn the data abstractor when the answer to variable **Date of operation** is before the **Admission date** or after the **Discharge date**.

On the **Procedures** screen, select the **Edit Rules** mode, and click on variable **Date of operation**.

Select rule type “After Entry Of Me Warn If...”

On the **Edit Rules** window, click on the right cell and select "**Variable**" and select variable “**Admission date**" from the **Select Variable** window.

On the second row, click on right cell and select "**Variable**" and select variable “**Discharge date**."

Save the rule.

The English translation of the rule should read:

"**After entry of 'Date of operation [TSTOPDT]', give warning IF.. 'Date of operation' < 'Admission Date' OR 'Date of operation' > 'Discharge Date'"**

Return to the **Laboratory** screen and set a rule to warn the data abstractor when the answer to variable **Hemoglobin at discharge** (**TSTHEMOG**) is less than 8 or more than 16.

After selecting rule type “**After Entry Of Me Warn If...**” and accessing the **Edit Rules** window, click on right cell and select "**Value**" and type “**8**."

On the second row, click on the right cell and select "**Value**" and type “**16**."

The English translation of the rule should read:

"**After entry of 'Hemoglobin at discharge [TSTHEMOG]', give warning IF.. 'Hemoglobin at discharge' < 8 OR 'Hemoglobin at discharge' > 16**"

Data Entry Rules for Advanced User

Create Skip Patterns

This example illustrates that a variable can be enabled/disabled on one screen based on the information entered for a variable on a different screen.

Disable variable **Hemoglobin at discharge** on the **Laboratory** screen if variable **Discharge status** on the **Discharge** screen is "**Expired**."

Select the **Discharge** screen and select the **Edit Rules** mode, click on variable **Discharge status** and select rule type "**On entry of me disable ? if ...**" from the *Rules* box.

On the **Edit Rule** window, select the <DISABLE ? VARIABLE ?> button, select the **Laboratory** screen and the "**Hemoglobin at discharge**" variable from the lists provided.

On the **Edit Rules** window, click on value in the right cell and select "**Expired**."

Enable variable **Indicate the status of operation** if variable **Date of operation** is earlier than the admission date or later than the discharge date.

On the **Procedures** screen, select the **Order/ Indent/Unindent Variables** function. On the **Order/Indent/Unindent** window, make variable **Status of operation** a child of variable **Date of operation** by indenting it. Click on variable **Status of operation**. Click on the <ENABLE IF> button.

On the **Edit Rules** window, click on the right cell and select "**variable**."

On the **Select Variable** window, select the **Procedure** screen and the variable "**Admission date**".

Clear row 2 by clicking anywhere on row 2 (pressing the <ESC> key to escape from popup if necessary) and clicking on the <CLEAR ROW> button.

Build Exit Logic

Create the logic to display an exit case warning message when the data abstractor enters "**Expired**" for variable **Discharge status**."

On the **Discharge** screen, select the **Edit Rules** mode, click on variable **Discharge status** and select rule type "**Exit Case if...**".

On the **Edit Rules** window, click on value in the right cell and select "**Expired**."

Build Save To Variable Logic

Create the logic to save "0" to variable **Lowest pulse** on the **Procedures** screen when the data abstractor enters "**Expired**" for variable **Discharge status**.

On the **Discharge** screen, select the **Edit Rules** mode, click on variable **Discharge status** and select rule type "**After entry of me save..? to ? if**."

On the **Edit Rules** window, select the "**Save..?_ variable_ ?**" function.

On the **Select Variable, Value To Save** window, select the **Procedures** screen and variable **Lowest pulse**. Click on Get Value and enter "**0**" as the lowest pulse value.

On the **Edit Rules** window, click on value in the right cell and select “**Expired**."